

FIG. 1 Wavelet Tiling of an  $N$ -Point Digital  $t$ - $f$  Space

**FIG. 2 Wavelet Iterated Filter Bank for Tiling t-f Space in FIG. 1**

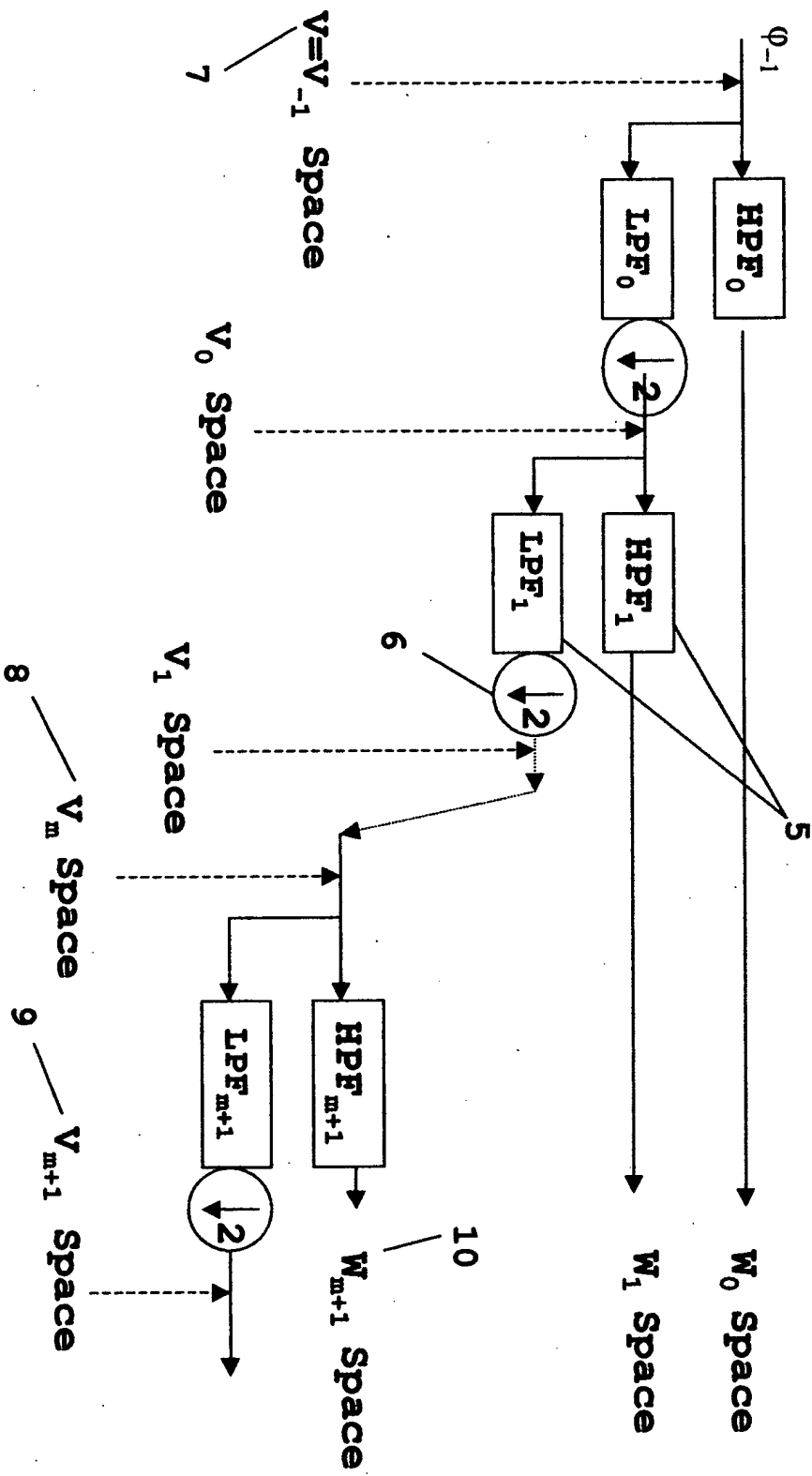
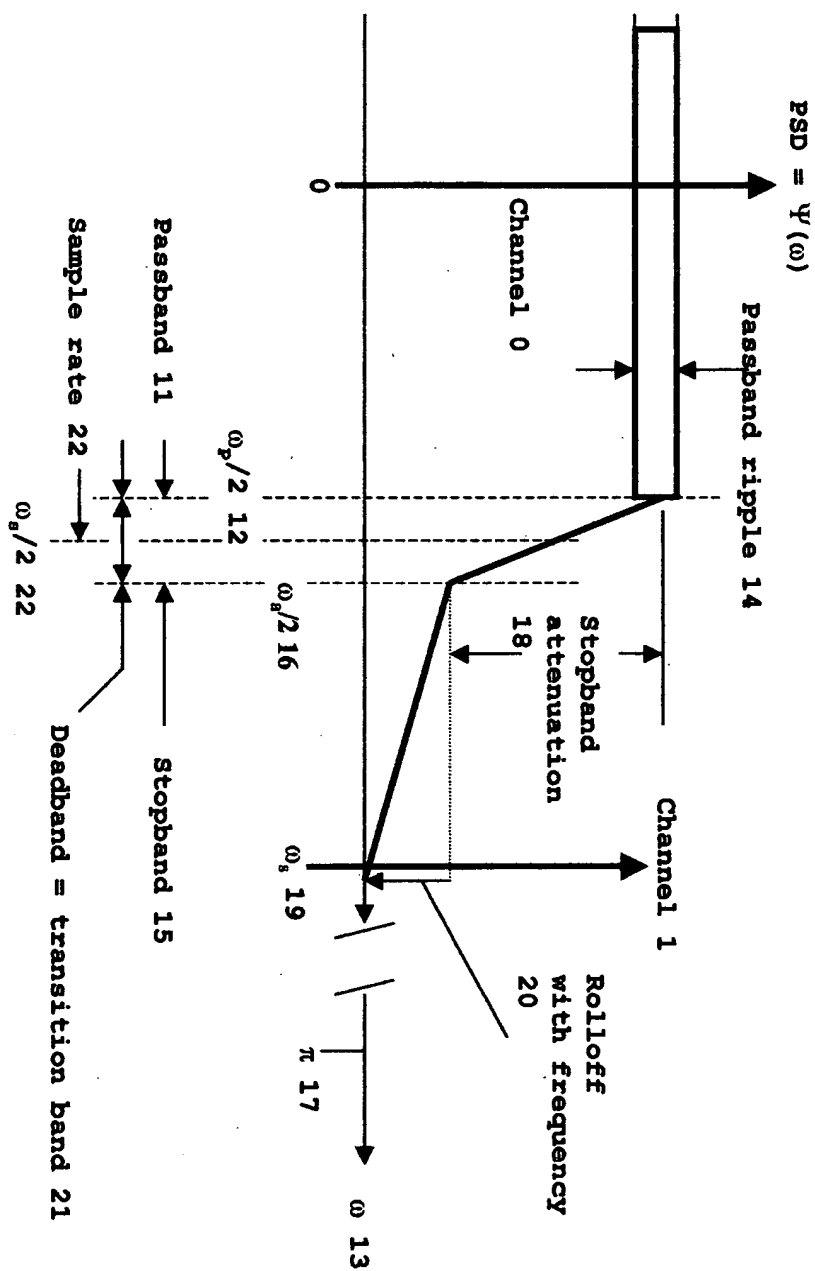
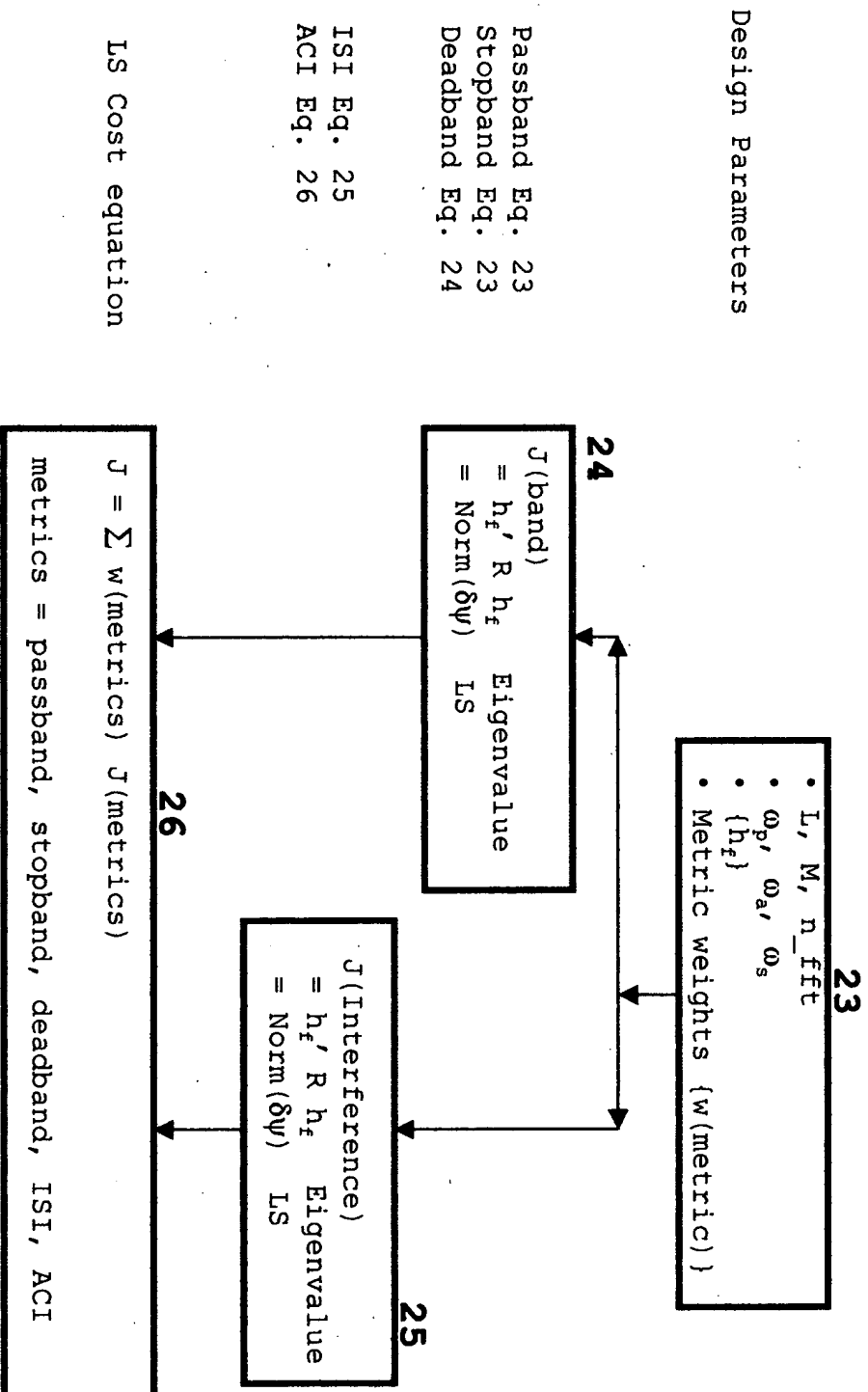


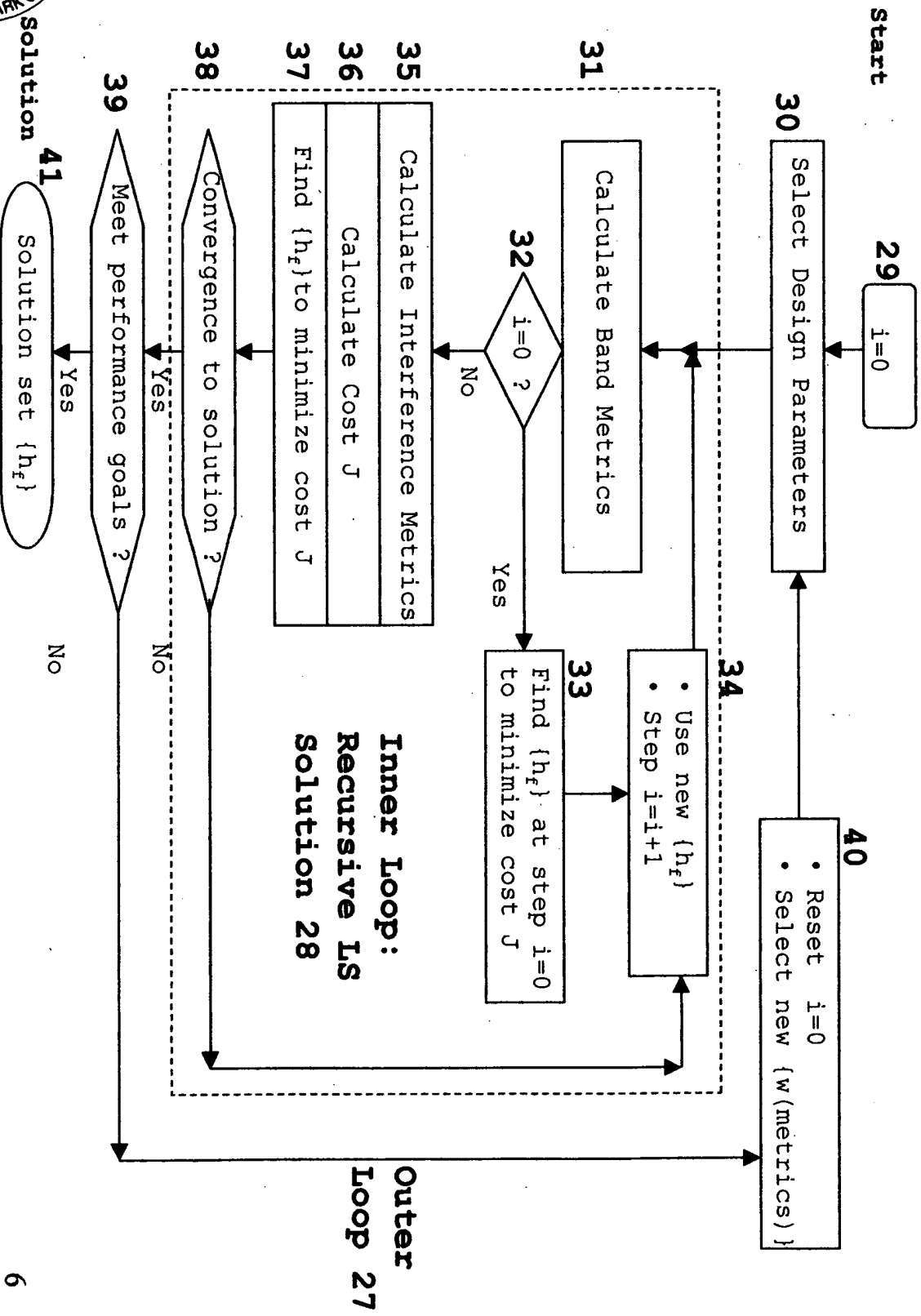
FIG. 3 PSD Requirements for Communications



**FIG. 4 IS Metrics and Cost Function**



**FIG. 5 IS Recursive Solution Algorithm**



# Replacement Sheet

FIG. 5A LS RECURSIVE DESIGN ALGORITHM IN MATLAB

## 5.0 CODE TO DESIGN:

- MOTHER WAVELET IN FIG. 6
- NEW WAVELET FROM MOTHER WAVELET
- PERFORMANCE DATA AND PLOTS

29

### % STEP 1 DESIGN PARAMETERS

% STEP 1.1 SCENARIO PARAMETERS

```
%=====
M      = 16;           % Wavelet sample interval
L      = 16;           % nominal Wavelet length in units of M
N      = M*L+1;        % Wavelet length N=ML+1
fs     = 1;            % normalized channel spacing
fp     = 0.8864;        % = normalized Wavelet sample rate
nf      = 16;           % number of design harmonics
n_fft  = 1024;          % FFT size for spectrum centered at 0
ebno   = 6.0;           % dB, Eb/No
x_imbal_aci = 6.0;      % dB, channel-to-channel imbalance
%=====
```

### % STEP 1.2 DERIVED PARAMETERS

```
twopl  = pi*2;          % definition
nc      = M;            % maximum number of channels allowed
nfft_wsr = (n_fft/M) * 0.5 * Wavelet sample rate
f_pass  = fp/(M*fs)      % wp/2pi edge of passband
f_stop  = (2-fp)/(M*fs)  % ws/2pi edge of stopband
nfft_pass = floor(f_pass*n_fft) % edge of passband
%=====
% STEP 1.3 OPTIMIZATION PARAMETERS
```

```
%=====
n_iteration = 10;       % number of iterations for LS design
alpha_1     = 1.e-2     % weighting for passband
alpha_2     = 0.80      % weighting for stopband
alpha_3     = 2.e-3     % weighting for ISI
alpha_4     = 0.5       % weighting for ACI
alpha_5     = 0         % weighting for deadband
%=====
```



# Replacement Sheet

FIG. 5B

30

## % STEP 2 INITIALIZATION CALCULATIONS

### % STEP 2.1 WAVELET LENGTH PARAMETERS

```

=====
%====
nodd=fix(N/2);
nodd=N-2*nodd;
if(nodd==1)
    m=(N-1)/2 % N is odd
    nrow=m+1;
else
    m=N/2; % N is even
    nrow=m;
end

```

### % STEP 2.2 MATRIX "bw\_matrix" MAPS WAVELET FREQUENCY DESIGN HARMONICS INTO WAVELET TIME RESPONSE

```

%====
bw_matrix=zeros(m,n_f);
for i_x=1:m
    % freq
    ang=2*pi*i_x*rem((1_x)*(0:n_f-1)/(N-1),1); % time
    bw_matrix(1_x+1,:)=2*cos(ang);
end

```

```

bw_matrix(1,:)=ones(1,n_f);
%====
% STEP 2.3 FUNCTION "pmm" CALCULATES PASSBAND, STOPBAND IS
% ERROR MATRICES FOR THE METRICS J(PASS), J(STOP) IN
% EQ. 23 AND FUNCTION "pmm_d" CALCULATES ERROR MATRIX
% FOR J(DEAD) IN EQ. 24

```

### % STEP 2.4 MATRIX "c\_matrix" USED FOR ISI, ACI IS ERROR METRICS J(ISI) IN EQ. 25 AND J(ACI) IN EQ. 26

```

%====
au=eye(m+1,m+1); % identity matrix
bd=rot90(au); % rotation
c_matrix(bd); flipud(bd(1:m,1:(m+1)))/2;
c_matrix(m+1,1)=c_matrix(m+1,1)*2;

```

### % STEP 2.5 PASSBAND, STOPBAND, WAVELET SAMPLE RATE TEMPLATES

```

%====
%====set up passband and stopband template
v_1=1:nfft_pass+1;
v_2=nfft_pass+2:nfft_stop;
v_3=nfft_stop+1:nfft_stop+nfft_pass;
hw_ref=zeros(size(v_1))-110*ones(size(v_2)) ...
    zeros(size(v_3));
%====set up wavelet sample rate template
v_1b=1:nfft_wsr;
v_2b=nfft_wsr+1:nfft_wsr+1;
v_3b=nfft_wsr+2:nfft_wsr+nfft_pass+nfft_stop;
hw_wsr=[-110*ones(size(v_1b)) zeros(size(v_2b)) ...
    -110*ones(size(v_3b))];
%====

```



# Replacement Sheet

FIG. 5c

31

%STEP 3 PASSBAND, STOPBAND, DEADBAND LS ERROR MATRICES

```
% STEP 3.1 J(PASSBAND) LS ERROR MATRIX "passband"
```

```
%=====
```

```
omega_l = 0.0 * pi;
```

```
omega_u = f_pass * pi ; % 0.0554
```

```
an=ones(1,nrow);
```

```
passband = pmn(omega_l, omega_u, N, an) ;
```

```
%=====
```

```
% STEP 3.2 J(STOPBAND) LS ERROR MATRIX "stopband"
```

```
%=====
```

```
omega_l = f_stop * pi; %
```

```
omega_u = pi ;
```

```
an=zeros(1,nrow);
```

```
stopband = pmn(omega_l, omega_u, N, an) ;
```

```
%=====
```

```
% STEP 3.3 J(DEADBAND) LS ERROR MATRIX "deadband"
```

```
%=====
```

```
omega_l = f_pass * pi; %
```

```
omega_u = f_stop * pi;
```

```
an=ones(1,nrow);
```

```
%deadband = pmn(d(omega_l, omega_u, N, an) ;
```

```
%deadband = zeros(nrow,nrow);
```

```
%=====
```

```
% STEP 3.4 WEIGHTED LS ERROR MATRIX "p_total" FOR THE WEIGHTED
```

```
% SUM OF J(PASSBAND), J(STOPBAND), J(DEADBAND)
```

```
%=====
```

```
p_total= alpha_1*passband+alpha_2*stopband+alpha_5*deadband;
```

```
%=====
```

```
% STEP 3.5 CONVERT LS ERROR MATRIX IN TIME "p_total" TO LS
```

```
% ERROR MATRIX IN FREQUENCY "pw_t"
```

```
%=====
```

```
pw_total = bw_matrix'*(p_total*bw_matrix);
```

```
pw_t = pw_total;
```

```
%=====
```

32

% STEP 4 ITERATIVE EIGENVALUE SOLUTION

```
%=====
```

```
hn_data = [];
```

```
hw_data = [];
```

```
err_ls = [];
```

```
loss_ls = [];
```

```
for i_iteration = 1:n_iteration
```

```
    i_iteration
```

```
%=====
```

```
%=====
```



# Replacement Sheet

FIG. 5D

```

=====
% STEP 4.1 FOR EACH ITERATION "i_iteration" FIND EIGENVECTOR
% IN FREQUENCY THAT MINIMIZES THE COST FUNCTION J IN
% EQ. 27 WHOSE IS ERROR MATRIX IS "pw_t"
%====
eig_val = eig(pw_t);
[eig_vec eigval] = eig(pw_t);
[eigval_min,min_index] = min(eig_val);
=====
% STEP 4.2 MAP EIGENVECTOR INTO:
% - WAVELET FREQUENCY DESIGN HARMONICS "hw_eig"
% - WAVELET IMPULSE RESPONSE IN TIME "hn"
%====
b_vector = bw_matrix * eig_vec(:,min_index);
hw_eig = eig_vec(:,min_index);
hw_eig(1) = 2*hw_eig(1);
hw_max = max(hw_eig);
hw_eig = hw_eig/hw_max;
if (nodd == 1) % N is odd
    hn(1:m) = 0.5*b_vector(m+1):-1:2;
    hn(m+1) = b_vector(1);
    hn(m+2:N) = hn(m:-1:1);
else
    hn(m:-1:1) = 0.5 * b_vector(1:m);
    hn(m+1:1:2*m) = hn(m:-1:1);
end % nodd
hmax = max(abs(hn));
scale_wv = 1. / (hmax^2);
% normalized hn is the normalized Wavelet response
hn = hn/hmax;
=====
% STEP 4.3 PASSBAND RIPPLE "xripple" AND
% STOPBAND ATTENUATION "xstop"
%====
% Fourier transform of hn & hn in the next channel
ich = 0;
arg_rot = twopi*rem( (0:N-1)*ich/nc , 1 );
[freq, hw_db] = freq_rsp(hn, arg_rot, n_fft);
% hn_data = [hn_data hn'];
% hw_data = [hw_data hw_db'];
%====
%==== peak_to_peak ripple in passband
max_ripple = max( hw_db(1: nfft_pass+1));
min_ripple = min( hw_db(1: nfft_pass+1));
xripple = max_ripple - min_ripple;
%==== stopband attenuation
xstop = max(hw_db(nfft_stop+1:nfft_stop+nfft_pass+1));
=====

```



# Replacement Sheet

FIG. 5E

```

33
% STEP 5 WEIGHTED ISI ERROR METRICS FOR:
%
% - J(PASSBAND) = "beta_pass"
% - J(STOPBAND) = "beta_stop"
% - J(DEADBAND) = "beta_dead"
%
%=====
err_pass = b_vector' * passband * b_vector;
err_stop = b_vector' * stopband * b_vector;
err_dead = b_vector' * deadband * b_vector;
beta_pass = alpha_1 * err_pass;
beta_stop = alpha_2 * err_stop;
beta_dead = alpha_5 * err_dead;
%=====

34
% STEP 6 ISI AND ACI ISI:
%
% - MATRICES "w_matrix" AND "w_f_matrix"
% - METRICS J(ISI)="errm_isi" AND J(ACI)="errm_aci"
% - SNR ERROR CONTRIBUTORS "errv_isi" AND "errv_aci"
%
%=====
% STEP 6.1 J(ISI):
%
% - LS ERROR MATRIX "w_matrix"
% - J(ISI) = "errm_isi"
% - SNR LOSS ISI ERROR "errv_isi"
%
%=====
for k_wave = 0:M
    n_1 = N - 1 - k_wave*nc;
    w_vector(k_wave+1) = 0.;
    for i1 = 0:n_1
        % ISI error residual vector w_vector
        w_vector(k_wave+1) = w_vector(k_wave+1) + hn(i1+1)*hn(i1+1+
            nc*k_wave);
    end
end
%
scale_isi_aci = 1/w_vector(1);
w_vector = w_vector * scale_isi_aci; % normalize
errv_isi = sum(w_vector(2:M) .* w_vector(2:M)); % ISI LS error
% 2-sided power summation of isi residual errors
errv_isi = 2. * errv_isi;
errv_isimax = max(abs(w_vector(2:M)));
%=====
a_matrix = zeros(m+1, 2*m+1);
for i_r = 0:m
    n_cc = 1_r * nc;
    if (i_r >= 1 & i_r <= M)
        nic = (n_cc+1):(2*m+1);
        a_matrix(i_r+1, nic) = hn(nic - n_cc);
    end
end
%=====

```

# Replacement Sheet

FIG. 5E

```

33
% STEP 5 WEIGHTED ISI ERROR METRICS FOR:
%
% - J(PASSBAND) = "beta_pass"
% - J(STOPBAND) = "beta_stop"
% - J(DEADBAND) = "beta_dead"
%
=====
err_pass = b_vector' * passband * b_vector;
err_stop = b_vector' * stopband * b_vector;
err_dead = b_vector' * deadband * b_vector;
beta_pass = alpha_1 * err_pass;
beta_stop = alpha_2 * err_stop;
beta_dead = alpha_5 * err_dead;
%
=====

34
% STEP 6 ISI AND ACI IS:
%
% - MATRICES "w_matrix" AND "w_f_matrix"
% - METRICS J(ISI) = "errv_isi" AND J(ACI) = "errm_aci"
% - SNR ERROR CONTRIBUTORS "errv_isi" AND "errv_aci"
%
=====
% STEP 6.1 J(ISI):
%
% - IS ERROR MATRIX "w_matrix"
% - J(ISI) = "errv_isi"
% - SNR LOSS ISI ERROR "errv_isi"
%
=====
for k_wave = 0:M
    n_1 = N - 1 - k_wave*nc;
    w_vector(k_wave+1) = 0.;
    for l1 = 0:n_1
        % ISI error residual vector w_vector
        w_vector(k_wave+1) = w_vector(k_wave+1) + hn(l1+1)*hn(l1+1) + ...
            nc*k_wave);
    end
end
scale_isi_aci = 1/w_vector(1);
w_vector = w_vector * scale_isi_aci; % normalize
errv_isi = sum(w_vector(2:M) .* w_vector(2:M)); % ISI IS error
% 2-sided power summation of isi residual errors
errv_isi = 2. * errv_isi;
errv_isiMax = max(abs(w_vector(2:M)));
%
a_matrix = zeros(m+1, 2*m+1);
for l_r = 0:m
    n_cc = l_r * nc;
    if (l_r == 1 & l_r == M)
        n1c = (n_cc+1):(2*m+1);
        a_matrix(l_r+1, n1c) = hn(n1c - n_cc);
    end
end
%
=====

```

# Replacement Sheet

FIG. 5G

35

```
% STEP 7 - WEIGHTED LS ERROR METRICS
% - UPDATE LS ERROR MATRIX "pw_t" FOR NEXT
% ITERATION
```

```
=====
% STEP 7.1 WEIGHTED LS ERROR METRICS FOR ISI, ACI, TOTAL
% - WEIGHTED ISI LS ERROR METRIC "beta_1st"
% - WEIGHTED ACI LS ERROR METRIC "beta_ac1"
% - TOTAL LS ERROR METRIC J = "errm_ls"
```

```
beta_1st = alpha_3*errm_1st;
beta_ac1 = alpha_4*errm_ac1;
errm_ls = beta_pass+beta_stop+beta_dead+beta_1st+beta_ac1;
```

```
% STEP 7.2 SAVE WEIGHTED LS ERROR METRICS FOR EACH ITERATION
```

```
if i_iteration==1
    scale_err = errm_ls;
```

```
end
beta_pass_1 = beta_pass * 1./errm_ls; % in fraction
beta_stop_1 = beta_stop * 1./errm_ls; % in fraction
beta_dead_1 = beta_dead * 1./errm_ls; % in fraction
beta_1st_1 = beta_1st * 1./errm_ls; % in fraction
beta_ac1_1 = beta_ac1 * 1./errm_ls; % in fraction
errm_ls = errm_ls / (alpha_1+alpha_2+alpha_3+alpha_4+alpha_5);
errm_ls = errm_ls / scale_err;
```

```
err_ls = [err_ls; i_iteration];
beta_pass_1 beta_stop_1 beta_dead_1 beta_1st_1 beta_ac1_1 errm_ls];
```

```
% STEP 7.3 UPDATE J LS ERROR MATRIX "pw_t" FOR NEXT ITERATION
```

```
p_t = p_total+ alpha_3 * w_matrix+ alpha_4 * w_f_matrix;
pw_t = bw_matrix*p_t* bw_matrix;
```

36

```
% STEP 8 SIGNAL-TO-NOISE SNR LOSS
```

```
% - PASSBAND RIPPLE LOSS "xloss_ripple", dB
% - ISI LOSS "xloss_1st", dB
% - ACI LOSS "xloss_ac1", dB
% - TOTAL LOSS "xloss_total", dB
```

```
% STEP 8.1 SNR LOSSES DUE TO PASSBAND RIPPLE, ISI, ACI, AND
% THE TOTAL SNR LOSS
```

```
passband_ripple_loss = 10.*log10(1./20.);
x_delta = 10.*log10(1./20.) - 1.;
xloss_ripple = -10.*log10(1.0 - x_delta^2);
```

```
isi_loss = 10.*log10(1./20.);
xebno = 10.*log10(1./20.);
xx_1st = xebno * errm_1st;
xloss_1st = 10.*log10(1.0 + xx_1st);
```



# Replacement Sheet

FIG. 5H

```

=====
ac1_loss
x_g_act = 10.^( x_lmbd_act / 10. ) ;
xx_act = xebno * errv_act * x_g_act ;
=====
xloss_act = 10. * log10( 1. + xx_act ) ;
=====
total_loss
xloss_total = 10. * log10( 1. + xx_lsl + xx_act ) + xloss_ripple ;
=====
STEP 8.2 SAVE SNR LOSSES FOR EACH ITERATION
=====
loss_ls = [loss_ls ; 1_iteration xloss_total ....
           xloss_ripple xloss_lsl xloss_act ] ;
=====
===== end of iterations
end
=====

```

37

STEP 9 WAVELET DESIGN FOR FIG. 6

STEP 9.1 WAVELET FREQUENCY DESIGN HARMONICS "hn\_elg"

```

=====
'Harmonic number Harmonic value'
[(0:n_f-1)' hn_elg]
=====

```

0	0.9651
1.0000	0.9499
2.0000	0.9842
3.0000	0.9485
4.0000	0.9869
5.0000	0.9434
6.0000	1.0000
7.0000	0.8428
8.0000	-0.2266
9.0000	-0.0018
10.0000	0.0019
11.0000	-0.0006
12.0000	0.0003
13.0000	0.0001
14.0000	-0.0000
15.0000	0.0002

STEP 9.2 WAVELET TIME RESPONSE "hn"

```

=====
'Sample index Wavelet response'
[(0:m)' hn(m+1:2*m+1)']
=====

```

0	1.0000
1.0000	0.9941
2.0000	0.9765
3.0000	0.9476
4.0000	0.9080
5.0000	0.8586



# Replacement Sheet

FIG. 5I

6.0000	0.8004
7.0000	0.7347
8.0000	0.6627
9.0000	0.5860
10.0000	0.5062
11.0000	0.4248
12.0000	0.3434
13.0000	0.2635
14.0000	0.1867
15.0000	0.1141
16.0000	0.0471
17.0000	-0.0135
18.0000	-0.0666
19.0000	-0.1117
20.0000	-0.1485
21.0000	-0.1766
22.0000	-0.1961
23.0000	-0.2073
24.0000	-0.2106
25.0000	-0.2065
26.0000	-0.1959
27.0000	-0.1795
28.0000	-0.1583
29.0000	-0.1335
30.0000	-0.1060
31.0000	-0.0769
32.0000	-0.0474
33.0000	-0.0182
34.0000	0.0095
35.0000	0.0350
36.0000	0.0577
37.0000	0.0769
38.0000	0.0923
39.0000	0.1035
40.0000	0.1106
41.0000	0.1134
42.0000	0.1121
43.0000	0.1071
44.0000	0.0987
45.0000	0.0873
46.0000	0.0736
47.0000	0.0581
48.0000	0.0414
49.0000	0.0242
50.0000	0.0070
51.0000	-0.0096
52.0000	-0.0250
53.0000	-0.0389
54.0000	-0.0507
55.0000	-0.0603
56.0000	-0.0674
57.0000	-0.0719
58.0000	-0.0738
59.0000	-0.0731



# Replacement Sheet

FIG. 5J

60.0000	-0.0701
61.0000	-0.0649
62.0000	-0.0578
63.0000	-0.0492
64.0000	-0.0393
65.0000	-0.0287
66.0000	-0.0177
67.0000	-0.0066
68.0000	0.0041
69.0000	0.0141
70.0000	0.0231
71.0000	0.0309
72.0000	0.0372
73.0000	0.0420
74.0000	0.0451
75.0000	0.0465
76.0000	0.0463
77.0000	0.0445
78.0000	0.0414
79.0000	0.0370
80.0000	0.0315
81.0000	0.0253
82.0000	0.0185
83.0000	0.0113
84.0000	0.0042
85.0000	-0.0029
86.0000	-0.0095
87.0000	-0.0155
88.0000	-0.0208
89.0000	-0.0252
90.0000	-0.0287
91.0000	-0.0311
92.0000	-0.0325
93.0000	-0.0328
94.0000	-0.0321



# Replacement Sheet

FIG. 5K

113.0000	0.0187
114.0000	0.0179
115.0000	0.0168
116.0000	0.0154
117.0000	0.0138
118.0000	0.0121
119.0000	0.0103
120.0000	0.0085
121.0000	0.0067
122.0000	0.0051
123.0000	0.0036
124.0000	0.0024
125.0000	0.0013
126.0000	0.0006
127.0000	0.0001
128.0000	-0.0000

38

% STEP 10 ITERATION CONVERGENCE IS MEASURED BY THE  
% CONVERGENCE OF THE LS ERRORS IN

% figure(1), figure(2)

% plots

figure(1)

plot(err\_LS(:,1),err\_LS(:,7),'k')

legend('Total LS error relative to iteration=1')

ylabel('Total LS error relative to iteration=1')

xlabel('Iteration number')

title('TOTAL LS ERROR J VS. ITERATION')

grid on

hold on

%====

figure(2)

plot(err\_LS(:,1),err\_LS(:,2),'k')

hold on

plot(err\_LS(:,1),err\_LS(:,3),'k--')

plot(err\_LS(:,1),err\_LS(:,4),'k')

plot(err\_LS(:,1),err\_LS(:,5),'b')

plot(err\_LS(:,1),err\_LS(:,6),'b--')

title('LS ERROR CONTRIBUTORS VS. ITERATION')

legend('passband','stopband','deadband','ISI','ACT')

ylabel('LS error relative to total=1')

xlabel('Iteration number')

grid on

hold on





# Replacement Sheet

FIG. 5L

```

39
% STEP 11 PARAMETERS ARE SELECTED TO OPTIMIZE:
%
% - WAVELET FILTER PERFORMANCE IN figure(3)
% - WAVELET RIPPLE, ISI, ACI SNR LOSSES IN figure(4)
% - WAVELET TIME RESPONSE IN figure(5)
%=====
figure(3)
plot(freq*M, hw_db, 'k')
axis([0 200 -100 10])
grid on
xlabel('Frequency/Wavelet sample rate')
ylabel('Power Spectrum, dB')
hold on
x2=length(hw_ref);
x3=length(hw_wsr);
plot(freq(1:x2)*M, hw_ref, 'b-')
plot(freq(1:x3)*M, hw_wsr, 'b-')
legend('Wavelet response', 'pass & stop templates', 'Wavelet sample
rate')
title('WAVELET FREQUENCY RESPONSE')
grid on
axis([0 1.4 -100 0])
hold on
%=====
figure(4)
plot(loss_IS(:,1), loss_IS(:,2), 'k')
hold on
plot(loss_IS(:,1), loss_IS(:,3), 'k-')
plot(loss_IS(:,1), loss_IS(:,4), 'b')
plot(loss_IS(:,1), loss_IS(:,5), 'b-')
title('SNR LOSS VS. ITERATION')
legend('total', 'ripple', 'ISI', 'ACI')
ylabel('SNR LOSS, dB')
xlabel('Iteration number')
grid on
hold on
%=====
figure(5)
xx=(-m:m)';
xx=xx/M;
plot(xx, hn, 'k')
hold on
xlabel('Time/Wavelet sample rate')
ylabel('Wavelet time response')
hold on
title('WAVELET TIME RESPONSE')
grid on
axis([-8 8 -0.4 1])
%=====

```

# Replacement Sheet

FIG. 5M

```

40
% STEP 12 CALCULATION OF NEW WAVELET WAVEFORM "hn_new"
%
% FOR THE PARAMETERS:
%   - "p" SCALE (DILATION)
%   - "q" TRANSLATION
%   - "k" FREQUENCY
%
%=====
% Wavelet parameters
p = 2 % scale change or dilation
q = 2 % time translation
k = 3 % frequency translation
%
% STEP 12.1 WAVELET SAMPLE INTERVAL "M_new" AND LENGTH "N_new"
%=====
%== Wavelet sample interval M_new for:
%=====
%
% Case 1: Fix M_new = M and dilate sampling
%   hn = hn(n*2^p - q*M)
%   n_new = n*2^p
%   = n_p for n = n_0 + n_p*2^p
%
% Case 2: Fix sampling and dilate M_new = 2^p*M
%   hn = hn(n - q*M_new)
%   M_new = M*(2^p);
%=====
% N_new = M_new*L+1; % Wavelet length
%=====
% nodd= fix(N_new/2);
% nodd = N_new - 2 * nodd;
% if (nodd == 1)
%   m_new = (N_new - 1) / 2 % N is odd
% else
%   m_new = N_new/2; % N is even
% end
%=====
% STEP 12.2 MATRIX "bw_matrix_new" FOR MAPPING WAVELET
% FREQUENCY DESIGN HARMONICS INTO NEWWAVELET IMPULSE
% RESPONSE IN TIME
%=====
% bw_matrix_new = zeros(m_new,n_f);
% for i=1:m_new
%   ang = 2*pi*rem((1-i)/(0:n_f-1)/(N_new-1),1); % time
%   bw_matrix_new(i_r+1, :) = 2 * cos(ang);
% end
% bw_matrix_new(1,:) = ones(1,n_f);
%=====
% STEP 12.3 MAP WAVELET FREQUENCY DESIGN HARMONICS "hw_elg"
% INTO NEW WAVELET IMPULSE RESPONSE IN TIME "hn_new"
%=====
% hn_0 = hn_new without translations in time & frequency
% hw_elg2 = hw_elg;
% hw_elg2(1) = 0.5*hw_elg(1);
%=====

```

# Replacement Sheet

FIG. 5N

```

b_vector = bw_matrix_new * hw_elg2;
if ( nodd == 1 ) % N_new is odd
    hn_0(1:m_new) = 0.5*b_vector(m_new+1):-1:2);
    hn_0(m_new+1) = b_vector(1);
    hn_0(m_new+2:N_new) = hn_0(m_new:-1:1);
else
    hn_0(m_new:-1:1) = 0.5 * b_vector(1:m_new);
    hn_0(m_new+1:2*m) = hn_0(m_new:-1:1);
end % nodd
hmax_0 = max( abs(hn_0) );
%==== normalized hn_0 is new baseband Wavelet with q=k=0
hn_0 = hn_0/ hmax_0;
%==== hn_1 is hn_0 with translation in time q*M_new
for n=1:N_new+q*M_new
    if n <= q*M_new
        hn_1(n) = 0;
    else
        hn_1(n) = hn_0(n-q*M_new);
    end
end
%====
%==== hn_new is hn1 with translation in frequency by k
for n=1:N_new+q*M_new
    hn_new(n) = hn_1(n)*exp( 1*(2*pi*k*(n-1)/(M_new*L)) );
end
%====
% STEP 12.4 PLOT WAVELET TIME RESPONSE FOR:
%
% - MOTHER WAVELET "hn"
% - NEW WAVELET "hn_new" WITHOUT FREQUENCY TRANSLATION
%====
figure(6)
xx1 = (L/2)*(1-1/2^p)*M_new;
xx2 = (L/2)*(1+1/2^p)*M_new;
for n=1:N_new+q*M_new
    if n<xx1 | n>xx2
        hn1(n) = 0;
    else
        hn1(n) = hn(n-xx1+1);
    end
end
x_n = (1:N_new+q*M_new)/M_new;
x_n = x_n-1/2;
plot(x_n,hn1,'k')
hold on
plot(x_n,hn_1,'k--')
grid on
hold on
legend('MOTHER WAVELET','NEW WAVELET')
xlabel('Time/hn_new sample rate')
ylabel('Wavelet time response')
hold on
title('TIME RESPONSE FOR MOTHER, NEW WAVELETS')
%=====

```

# Replacement Sheet

FIG. 50

```
% STEP 12.5 PLOT WAVELET FREQUENCY RESPONSE FOR:
%
% - MOTHER WAVELET "hn"
% - NEW WAVELET "hn_new"
%
% vs. frequency/hn sample rate
% vs. frequency/hn_new sample rate
%
=====
figure(7) % vs. frequency/hn sample rate
ich = 0;
arg_rot = twopi*rem( (0:N_new-1)*ich /nc , 1 );
[freq, hw_db] = freq_rsp(hn, arg_rot, n_fft);
plot(freq*M, hw_db, 'k')
hold on
ich=k;
arg_rot = twopi*rem( (0:N_new-1)*ich /M_new , 1 );
[freq, hw2_db] = freq_rsp(hn_0, arg_rot, n_fft);
plot(freq*M, hw2_db, 'k--')
axis([0 8 -100 10])
grid on
legend('MOTHER WAVELET', 'NEW WAVELET')
xlabel('Time/hn sample rate')
ylabel('Wavelet time response')
hold on
title('POWER SPECTRUM OF MOTHER, NEW WAVELETS')
xlabel('Frequency/hn sample rate')
ylabel('Power Spectrum, dB')
%=====
%==== plot frequency response of hn, hn_new
% vs. frequency/hn_new sample rate
figure(8)
plot(freq*M_new, hw_db, 'k')
hold on
plot(freq*M_new, hw2_db, 'k--')
axis([0 8 -100 10])
grid on
legend('MOTHER WAVELET', 'NEW WAVELET')
xlabel('Time/hn sample rate')
ylabel('Wavelet time response')
hold on
title('POWER SPECTRUM OF MOTHER, NEW WAVELETS')
xlabel('Frequency/hn_new sample rate')
ylabel('Power Spectrum, dB')
%=====
```



# Replacement Sheet

FIG. 5P

41

% STEP 13 FUNCTIONS USED IN MATLAB PROGRAM

% STEP 13.1 FUNCTION "pmn" COMPUTES MATRIX FOR J(BAND) IN  
EQ. 23

```

=====
function p_matrix= pmn(omega_1,omega_u, N,an)
%
% compute the real, symmetric, and positive definite matrix
% input: omega_1: lower edge (radians)
% omega_u: upper edge (radians)
% N: filter length, an(:): 1xm column vector
% output
% p_matrix(n,m): a nxm real, symmetry and positive-definite matrix
=====
twopi = 2. * pi;
% check filter lenght is odd or even
nodd = fix(N/2);
if ( nodd == 1)
    m = (N-1)/2; % filter length 'N' is odd
else
    m = N/2; % N is even
end
if ( nodd == 1)
    for n=0:m
        for ml=0:m
            if ( ml == n )
                if ( n == 0)
                    p_matrix(n+1,ml+1)=1./pi*(an(n+1)*an(n+1)+0.5)*(omega_u-omega_1)-....
                    2.* an(n+1) * ( sin( n*omega_u ) - sin( n*omega_1 ) ) ....
                    /n + (sin(2.*n*omega_u) - sin(2.*n*omega_1))/(4.*n);
                else
                    p_matrix(n+1,ml+1)=1./pi*(an(n+1)-1.)*(an(n+1)-1.)*(omega_u-omega_1);
                end
            else
                if ( ml ~= 0 & n ~= 0)
                    p_matrix(n+1,ml+1)=1./pi*(an(n+1)*an(ml+1)*(omega_u-omega_1)-....
                    -(an(ml+1)*ml*( sin( n*omega_u ) - sin( n*omega_1 ) ) + ....
                    an(n+1) *n * ( sin(ml*omega_u) - sin(ml*omega_1) ) )/( ml*n ) + ( ...
                    (n+ml) * ( sin( (n-ml)*omega_u ) - sin((n-ml)*omega_1) ) ...
                    +(n-ml)*( sin( (n+ml)*omega_u ) -sin((n+ml)*omega_1)))/(2.*(n*n-ml*ml)));
                else
                    if ( n == 0)
                        p_matrix(n+1,ml+1)=1./pi*(an(n+1)-1.)*(an(ml+1)*(omega_u-omega_1)-....
                        ( sin(ml*omega_u ) -sin( ml*omega_1 ) ) /ml );
                    end
                    if ( ml == 0)
                        p_matrix(n+1,ml+1)=1./pi*(an(ml+1)-1.)*(an(n+1)*(omega_u-omega_1)-....
                        ( sin(n*omega_u ) -sin(n*omega_1) ) /n );
                    end
                end
            end
        end
    end
end % end of ml loop
end % end of n loop
else
    %=====

```

# Replacement Sheet

FIG. 5Q

```

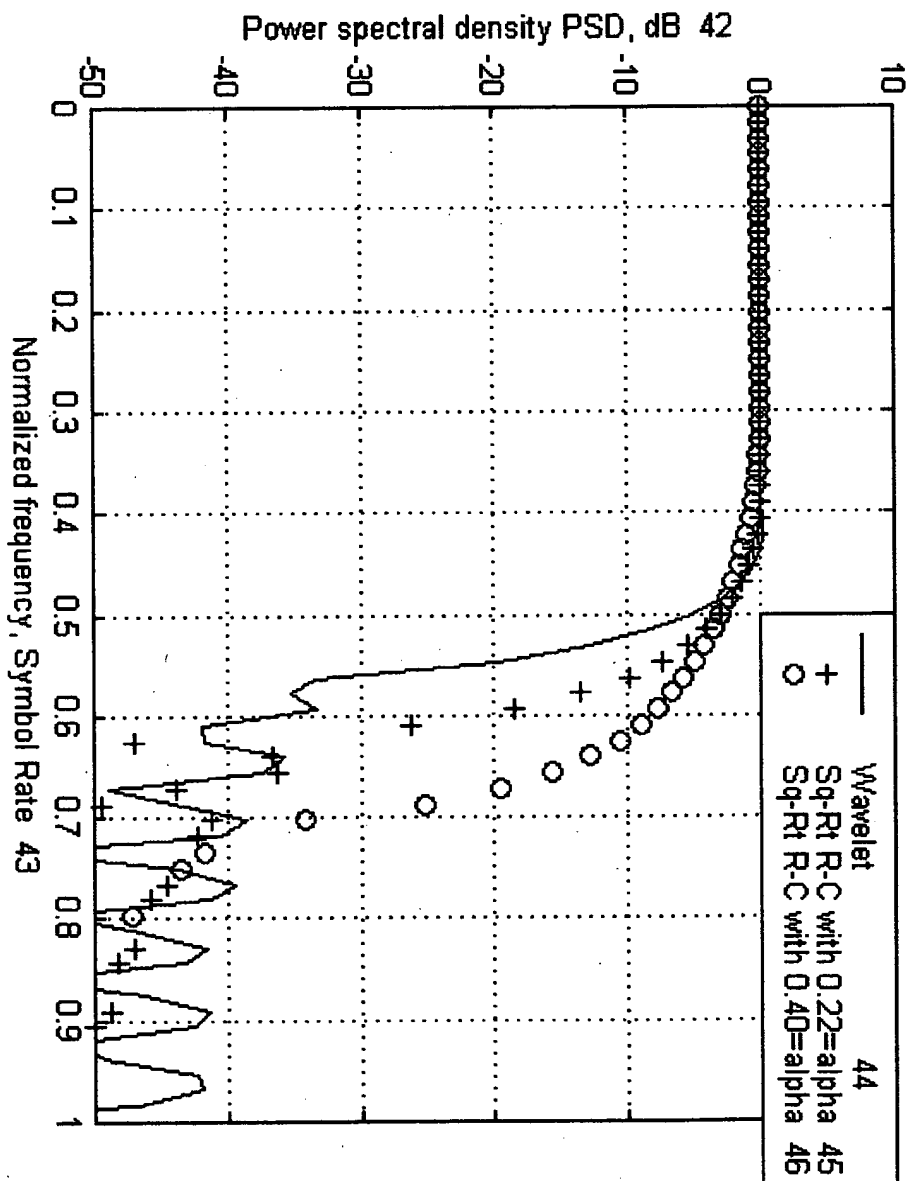
% ===== when N is even =====
for n = 0:m-1
    for m1 = 0:m-1
        if ( m1 == n )

            p_matrix(n+1,m1+1) = 1./pi * ( ...
                ( an(n+1)*an(n+1) + 0.5 ) * ( omega_u - omega_l ) - ...
                2. * an(n+1) * ( sin( (n+.5) * omega_u ) - ...
                sin( (n+.5) * omega_l ) ) / ( n + 0.5 ) + ...
                ( sin( (2*n+1) * omega_u ) - sin( (2*n+1) * omega_l ) ) ...
                / ( 2. * ( 2.*n + 1 ) ) ) ;

        else
            p_matrix(n+1,m1+1) = 1./pi * ( ...
                an(n+1) * an(m1+1) * ( omega_u - omega_l ) - ...
                an(m1+1) * ( sin( (n+.5) * omega_u ) - sin( (n+.5) * omega_l ) ) / ( n + 0.5 ) - ...
                an(n+1) * ( sin( (m1+.5) * omega_u ) - sin( (m1+.5) * omega_l ) ) / ( m1 + 0.5 ) + ...
                ( sin( (n-m1) * omega_u ) - sin( (n-m1) * omega_l ) ) / ( 2. * ( n-m1 ) ) + ...
                ( sin( ( n+m1+1) * omega_u ) - sin( ( n+m1+1) * omega_l ) ) / ( 2. * ( n+m1+1 ) ) ) ;
            end
        end % end of m1 loop
    end % end of n loop
end % end of if nodd = 1
% =====
% STEP 13.2 FUNCTION "freq_rsp" COMPUTES FOURIER TRANSFORM OF
% INPUT "hn" VS. FREQUENCY/WAVELET SAMPLE RATE
% =====
function [freq, hw_db] = freq_rsp(hn, arg_rot, n_freq)
% Fourier transform of input hn
% in normalized freq interval (0., 0.5)
% frequency response
% n_freq # of frequency
twopi = 2. * pi;
df = 0.5 / (n_freq - 1) ;
n_filter = length(hn);
m=(n_filter-1)/2;
freq = (0:df:0.5);
for nf = 1: n_freq
    arg=twopi * rem( freq(nf) * (1:n_filter) -1-m,1);
    hw = sum( hn .*exp( (-arg+arg_rot)*1 ) );
    hw_mag(nf) = abs( hw );
end
hw_max = max( abs(hw_mag) );
hw_mag = hw_mag /hw_max;
hw_db = 20. * log10( hw_mag+ 1.e-20);
% =====

```

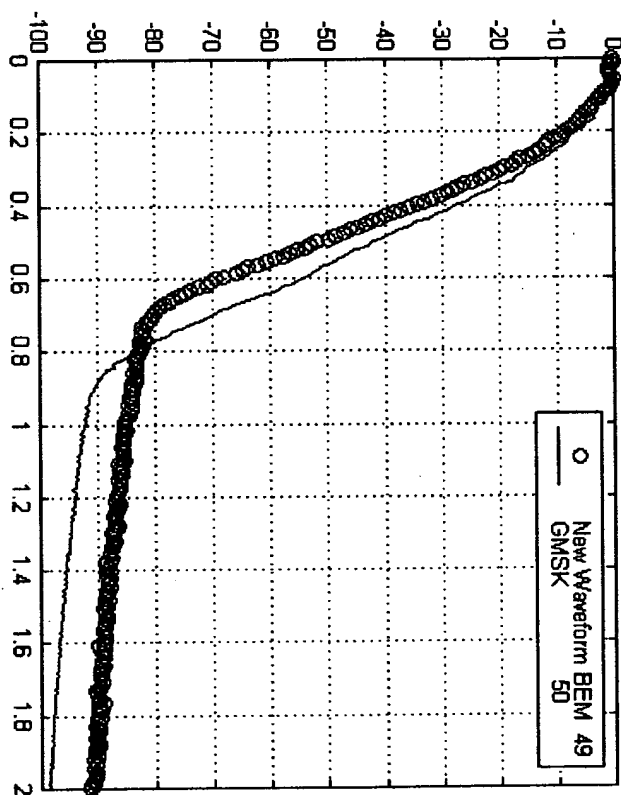
FIG. 6 PSD for Wavelet and Square-Root Raised Cosine





Power spectral density PSD, dB 47

FIG. 7 PSD for New Waveform BEM and GMSK



Normalized frequency, bit rate 48



**FIG. 8 Radar Ambiguity Functions of Wavelet  
 and Unweighted Chirp Waveform**

Wavelet Ambiguity Function 51

Unweighted Chirp Ambiguity Function 52

